Global Registration of Dynamic Range Scans for Articulated Model Reconstruction

WILL CHANG
University of California, San Diego
and
MATTHIAS ZWICKER
University of Bern

We present the articulated global registration algorithm to reconstruct articulated 3D models from dynamic range scan sequences. This new algorithm aligns multiple range scans simultaneously to reconstruct a full 3D model from the geometry of these scans. Unlike other methods, we express the surface motion in terms of a reduced deformable model and solve for joints and skinning weights. This allows a user to interactively manipulate the reconstructed 3D model to create new animations.

We express the global registration as an optimization of both the alignment of the range scans and the articulated structure of the model. We employ a graph-based representation for the skinning weights that successfully handles difficult topological cases well. Joints between parts are estimated automatically and are used in the optimization to preserve the connectivity between parts. The algorithm also robustly handles difficult cases where parts suddenly disappear or reappear in the range scans. The global registration produces a more accurate registration compared to a sequential registration approach, because it estimates the articulated structure based on the motion observed in all input frames. We show that we can automatically reconstruct a variety of articulated models without the use of markers, user-placed correspondences, segmentation, or template model.

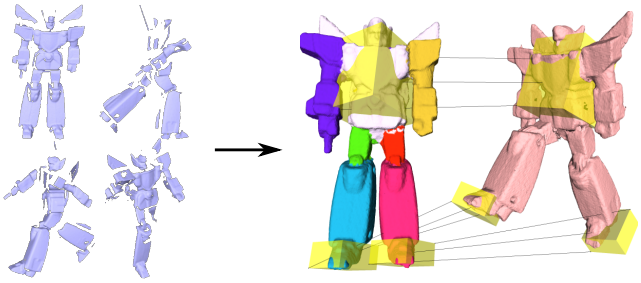
Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric Algorithms*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Surface Fitting*

General Terms: Algorithms, Measurement

Additional Key Words and Phrases: Range scanning, articulated model, non-rigid registration, animation reconstruction

ACM Reference Format:

Authors' addresses: land and/or email addresses.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© YYYY ACM 0730-0301/YYYY/11-ARTXXX $10.00
DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY



Input Range Scans — Poseable, Articulated 3D Model

Fig. 1. The articulated global registration algorithm can automatically reconstruct articulated, poseable models from a sequence of single-view dynamic range scans.

1. INTRODUCTION

While 3D scanning has traditionally focused on acquiring static, rigid objects, recent advances in real-time 3D scanning have opened up the possibility of capturing dynamic, moving subjects. Range scanning has become both practical and cost-effective, providing high-resolution, per-pixel depth images at high frame rates. However, despite the many advances in acquisition, many challenges still remain in the processing of dynamic range scans to reconstruct complete, animated 3D models.

Our research vision is to automatically reconstruct detailed, poseable models that animators can directly plug into existing software tools and use to create new animations. However, range scans have much missing data due to a limited view of a 3D subject from any single viewpoint at any point in time. To reconstruct a complete model, we must track the movement of the subject in each frame to align and integrate scans taken from different times and viewpoints. In addition, the reconstructed model should be easy to animate similar to how it actually moved in the range scans. Solving for a reduced set of parameters describing the surface motion allows us to meet this goal and improve the usability of the model.

We present a new method, articulated global registration, to address these challenges by reconstructing a rigged, articulated 3D model from dynamic range scans. Given a sequence of range scans of a moving subject, the algorithm automatically aligns all scans to produce a complete 3D model. We formulate our approach as a single optimization problem that simultaneously aligns partial surface data and recovers the motion model. This is accomplished without the assistance of markers, user-placed correspondences, a template, or a segmentation of the surface. Our method is unique because we perform the alignment by estimating the parameters of a reduced, articulated deformation model. In contrast to methods that focus

only on registration or reconstruction of the original recording, our method produces a 3D model that can be interactively manipulated with no further post-processing. Our main contributions are:

—A global registration algorithm for articulated shapes that optimizes the registration simultaneously over multiple frames,
—A novel registration formulation that produces a 3D model with skinning weights learned from incomplete examples,
—An improved robust registration technique to automate the global registration with initial pairwise alignments of adjacent frames.

Our method is useful because it does not require a template shape or a predefined skeletal structure, it handles range scans with fast motion and significant occlusion, and it produces a rigged 3D model. It is most useful when you cannot acquire a static pose of the subject. In this case, the object moves continuously, and scans at any point in time are incomplete. These situations are quite common unless you have many cameras to cover the surface completely. The articulated global registration is mainly applicable to articulated subjects. When applied to non-rigid datasets, it produces a rough piecewise rigid approximation of the surface motion.

We demonstrate the effectiveness of our algorithm by reconstructing several synthetic and real-world articulated models. Additionally, we present a simple extension of our algorithm to interactively manipulate the resulting 3D model.

2. RELATED WORK

Template-Based Reconstruction. A popular approach to reconstruct deforming sequences of range scans is to acquire a template shape and fit it to the scan data. Some methods rely on tracked marker locations to fit the template [Allen et al. 2002; 2003; Anguelov et al. 2005; Pauly et al. 2005; Bradley et al. 2008], while others utilize global optimization [Anguelov et al. 2004] or high frame rate scan sequences [Zhang et al. 2004; Weise et al. 2009]. Using a template provides many advantages in tracking and fitting the data, with the expense of requiring the user to scan or model it in advance. Our work addresses the more general problem of reconstructing a model without using a template.

Templates are also used for estimating shape using multiview silhouette/video data [de Aguiar et al. 2008; Vlasic et al. 2008; Gall et al. 2009] or sparse marker data [Park and Hodgins 2006; 2008]. Although these methods address the same problem of capturing deformable geometry, they do not address how to process high-resolution range scan data taken from just one or two views.

Most template based methods tend to recover surface detail only from the template and not from the input data. Works by Ahmed et al. [2008] and by Li et al. [2009] show that additional surface details can be extracted from the input data.

The work by Li et al. [2009] is most similar to our method, although it requires scanning a coarse template of the subject in advance. Other than the use of a template, their method assumes non-rigid deformations, whereas we assume an articulated (piecewise rigid) deformation model.

The difference between articulated global registration and more general non-rigid registration algorithms is that the rigidity of surface parts helps to improve the convergence of the registration when there is fast motion between scans. This fact was first pointed out by Huang et al. [2008], who show that rigid clustering of the surface improves the convergence of non-rigid registration. We also demonstrate this in our work (Figure 18) by comparing with the more general non-rigid registration of Wand et al. [2009]. While the non-rigid registration fails to converge when there is large motion, the articulated registration produces an accurate registration.

Templateless Reconstruction. To tackle the reconstruction problem without a template, many researchers have considered modeling a dynamic range scan sequence as a surface in four-dimensional space and time, rather than a single 3D surface that changes its configuration over time. Mitra et al. [2007] use kinematic properties of this 4D space time surface to track points and register multiple frames of a rigid object. Süßmuth et al. [2008] and Sharf et al. [2008] explicitly model and reconstruct the 4D space-time surface using an implicit surface representation. However, these techniques require the surface to be sampled densely in both space and time, which is an assumption that our method does not require. In addition, the method of Sharf et al. [2008] does not track points to produce correspondences between frames; it is more appropriate for filling in missing surface data not observed by the scanner. Recently Popa et al. [2010] reconstruct globally consistent mesh animations from video based capture sequences. In contrast, our method does not require video sequences nor does it rely on optical flow to compute correspondences between frames.

The algorithm by Wand et al. [2009] reconstructs an animated 3D model from range scan sequences without using a template. Compared to the works mentioned above, this method is more robust to missing data in the scans. It aligns multiple frames by solving the surface motion in terms of an adaptive displacement field. This motion representation handles smooth deformations well, but our representation is more compact and accurate for representing articulated motion. Wand et al. [2009] align and merge pairs of adjacent frames in a hierarchical fashion, gradually building the template shape hierarchically as well. In contrast, we simultaneously align all frames at once using an explicit piecewise rigid deformation model. In addition, our method is more robust to large movements and produces a fully rigged, poseable 3D model, rather than just reconstructing the original recorded motion sequence.

Our method is partly inspired by the articulated motion capture and reconstruction method of Pekelny and Gotsman [2008]. However, this method requires the user to manually segment a range scan in advance, whereas we automatically solve for the segmentation using the motion observed in all frames.

The articulated shape-from-silhouette (ASFS) algorithm [Cheung et al. 2003] is also similar to our method, because it reconstructs an articulated 3D model without using a template. The source of the data is 3D points extracted from multi-view video, where the cameras cover the scene from nearly all angles. In contrast, we reconstruct articulated models from single-view (or double-view) range scan data that has much missing data. In addition, ASFS estimates body kinematics using a sequential approach that models the joints one by one. The datasets are acquired by, for example, asking the person to move his left arm while keeping the rest of the body still (to find the position of the left shoulder joint). We estimate body kinematics for all joints simultaneously, so we do not require a dataset with such constrained movement.

Unsupervised Pairwise Registration. While our method is designed for aligning multiple range scans, several methods for aligning a pair of scans are related to our work as well. A closely related work is the method by Chang and Zwicker [2009], which solves for the alignment between a pair of range scans by estimating the parameters of a reduced deformable model. A possibility is to apply this method directly for multiple scans, using a sequential pairwise registration and accumulation approach. However, in this case the correct articulated structure is not estimated properly, because it considers the movement in only two frames at a time. Also, un-

less a very high resolution is used, the grid-based representation of the weights cannot handle difficult topological cases with close or nearby surfaces. In contrast, we use a graph based approach to robustly handle multiple frames and bad topological cases.

The transformation sampling and optimization approach by Chang and Zwicker [2008] is used in our work to initialize the registration between pairs of adjacent frames. However, this technique is too slow to apply for an entire sequence of range scans. We improve the performance of this method by subsampling the geometry. Our use of a graph to represent the deformation model is related to the approach by Li et al. [2008] and [Sumner et al. 2007]. However, we solve for weights on the graph nodes, as opposed to solving for a separate affine transformation at each node. The method by Huang et al. [2008] also uses a graph, but they use it as an approximation of geodesic distances in order to extract a set of geodesically consistent correspondences. The use of geodesic distances can make this approach problematic when a large amount of surface data is missing.

Deformation Modeling from Examples. Our inverse kinematics system resembles that of FaceIK [Zhang et al. 2004] or MeshIK [Sumner et al. 2005], which extrapolate a set of examples to match user constraints. However, the deformation model that we produce is a parametric model that explicitly models parts and joints, as opposed to a data-driven method that blends a set of example meshes. Therefore, our interactive IK system does not use the original examples at run-time and only uses the reconstructed deformation parameters (skinning weights and joints) to pose the 3D model.

Our deformation modeling approach is closer to the work on example-based skinning [James and Twigg 2005; Kavan et al. 2008] and skeleton extraction [Anguelov et al. 2004; Schaefer and Yuksel 2007; de Aguiar et al. 2008]. However, while these approaches estimate the deformation parameters using a set of complete examples that are already in correspondence, we estimate them directly from incomplete range scan data.

3. ALGORITHM OVERVIEW

The goal of our algorithm is to align a set of dynamic range scans and express the surface motion using a reduced set of parameters. We pose this problem as a skinning problem: finding transformations per frame and weights per vertex. When we apply these transformations to each scan according to the weights, all scans should be aligned with each other. We expect the range scans to be in temporal order, so that there is sufficient overlap between frames to align the scans.

The basic structure of our method is shown in Algorithm 1. The core of our method is the articulated global registration (lines 3–10). The main idea is to optimize the transformations and weights simultaneously across all frames to align them to a common reference pose. The optimization operates on a central data structure we call the dynamic sample graph (DSG). The DSG is formed on a subset of points sampled from the input scans. We select the points such that they form a uniform sampling of the complete surface. The DSG is dynamic because it continually incorporates more surface data from new scans as they are added to the optimization. The main advantage of the DSG is that it removes redundancies in the input range scans and, hence, makes the global registration tractable.

To give an overview of Algorithm 1, we start by precomputing an initial pairwise registration for each pair of adjacent frames (line 2, Section 4). Next, we create the initial DSG (line 3, Section 6). Then, each frame is introduced into the global registration one at a time (lines 5–10). For each frame, we apply the initial pairwise registration (line 6, Section 6) which gives an initial alignment of the new frame to the surface registered so far. We then detect occlusions and disocclusions of surface parts in the new frame (line 7, Section 7) and optimize the transformations and weights to simultaneously align all frames (line 8, Section 5.3). Lastly we update the DSG (line 9, Section 6) to incorporate new samples from the new frame, and we move on to the next frame. After finishing the entire sequence, the final post-processing step is to resample the surface densely and reconstruct a mesh of the completed surface (line 11, Section 8.1).

Algorithm 1: ARTICULATED GLOBAL REGISTRATION

Data: A sequence of range scans $(F_0, \ldots, F_{n-1})$
Result: Dynamic sample graph of the completed surface, weights $\mathcal{W}$ for each sample point, rigid transformations $\mathcal{T}$ for all parts and frames

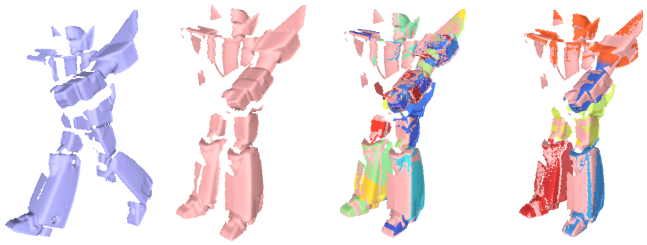
```
1  begin
2      Compute initial pairwise registration (Section 4);
3      Precompute DSG sample candidates (Section 6);
4      i ← 0;
5      while F_i ≠ F_{n-1} do
6          Apply pairwise registration of F_i, F_{i+1} (Section 6);
7          Detect occluded parts in F_{i+1} (Section 7);
8          Perform global registration of {F_0, ..., F_{i+1}}
           (Algorithm 2);
9          Update DSG (Section 6);
10         i ← i + 1;
11     Resample the DSG densely and reconstruct surface mesh
       (Section 8.1);
12     return DSG, W, T;
13 end
```

4. INITIAL PAIRWISE REGISTRATION

In a preprocessing step, we solve for an initial pairwise registration for each pair of adjacent frames. Since the scans have missing data and their poses can be far apart, the algorithm of Chang and Zwicker [2008] is well suited for producing a robust registration. It consists of two steps. First, we sample rigid transformations from feature-based correspondences between the scans. Second, we optimize the assignment of these transformations onto each vertex of the scans, such that applying the transformations produces an alignment of the scans while preserving their shape.

The details of the method are the same as originally described by Chang and Zwicker [2008]. However, with range scans that typically have tens of thousands of points, it is too slow to process an entire range scan sequence with many frames. To improve performance, we restrict the optimization to a small subset of points (typically a few thousand) sampled uniformly from each scan using best-candidate sampling [Mitchell 1991]. This makes sense for articulated movement, where the number of unique transformations producing the movement is small compared to the number of scanned points. We build a k-nearest neighbor graph with $k = 15$ on the subset of points to specify the smoothness constraints necessary for the optimization [Chang and Zwicker 2008].

After the optimization, we propagate the transformations assigned to the subset to all points using nearest neighbor interpolation. This produces the initial pairwise registration that we will use as an initialization for the global registration.



(a) Source (blue) and Target (red) 21325 points total (b) Alignment Using All Points: 1330.4 sec (c) Alignment Using 1000 Points: 87.3 sec

Fig. 2. Performance comparison for the initial pairwise registration. Optimizing on a subset of the points in (c) produces a similar registration as (b), but only takes a fraction of the time. The color variation in (b) and (c) visualizes how the transformations are assigned to the surface.

A comparison of the optimization using all points versus using a subset is shown in Figure 2. Although we obtain a good alignment in both cases, the improved method achieves a significant speedup.

5. GLOBAL REGISTRATION

The central part of our method is the optimization of transformations and weights that simultaneously align all frames. We first describe our deformation model in more detail.

5.1 Deformation Model

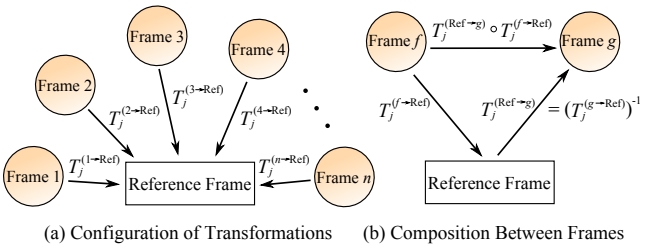
Transformations. We represent the surface motion using a set of rigid transformations in each frame. We designate the first frame as the *reference frame* and define the transformations relative to this reference [Neugebauer 1997]. Each transformation moves a part of the surface in frame f to align to the corresponding part in the reference frame. To aid our method, the user specifies a maximum number of rigid transformations B used to approximate the surface motion.

We use the notation $T_a^{(f\to\text{Ref})}$ to denote the ath transformation for frame f, which transforms in the direction *from* frame f *to* the reference frame (Figure 3a). Each $T_a^{(f\to\text{Ref})}$ transforms points $\mathbf{x}$ according to the formula $T_a^{(f\to\text{Ref})}(\mathbf{x}) = R_a^{(f\to\text{Ref})}\mathbf{x} + \vec{\mathbf{t}}_a^{(f\to\text{Ref})}$, where R is a rotation matrix and $\vec{\mathbf{t}}$ is a translation vector. We also express the relative transformation $T_a^{(f\to g)}$ between any two frames f, g by first transforming from frame f to the reference and then transforming from the reference to frame g (Figure 3b).

Weights. We associate the transformations to the points indirectly by assigning weights to each point. By changing the weights during the optimization, we can dynamically adjust where each transformation is being applied.

In our method, we solve for binary skinning weights. Thus, each point is associated with exactly one transformation. We take this approach because solving for smooth weights during registration leads to overfitting of both transformations and weights [Chang and Zwicker 2009]. Using binary weights essentially divides the points into *rigid parts*: sets of points that have the same weight.

Dynamic Sample Graph (DSG). We introduce the dynamic sample graph (DSG) to efficiently represent the skinning weights and make the optimization over all frames tractable. The DSG consists of sample points selected from each frame, edges between samples to form a graph, and a weight per sample. The goal of the optimization is to solve for weights and transformations that align the sample points to all frames simultaneously. We use the edges of the DSG to impose smoothness constraints when optimizing for the weights.



(a) Configuration of Transformations (b) Composition Between Frames

Fig. 3. (a) We solve for the set of transformations that align each input frame to the reference frame. (b) We can transform between any pair of frames f and g by first transforming from f to the reference and applying the inverse transformation to g.

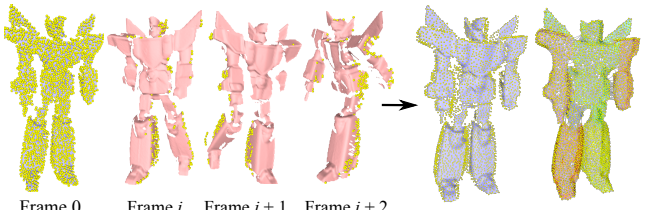


Fig. 4. Initially the vertices of the dynamic sample graph (DSG) are uniformly sampled from the reference frame. As we introduce more frames (i, $i+1$, $i+2$), we add samples only from parts of the surface that are missing in previous frames. On the right side of the arrow, we show all sample points and the DSG on the reference frame. Our sample selection strategy minimizes redundancy and yields a uniform sampling of the whole surface.

Not all sample points are from a single frame. Some are from frame 1, some from frame 2, and so on. As we incorporate each frame into the registration, we add samples to the DSG only from surface parts that are missing in all previous frames. At the same time, we select the samples to provide uniform and adequate coverage of the entire surface seen so far. This idea is illustrated in Figure 4. We discuss details on how to sample the points and form the edges of the DSG later in Section 6.

5.2 Optimization Objective

The optimization objective quantifies the quality of the alignment using three error terms: (1) $\mathcal{E}_{\text{fit}}(\mathcal{T},\mathcal{W})$, which measures the alignment distance between the DSG samples and all frames, (2) $\mathcal{E}_{\text{joint}}(\mathcal{T})$, which constrains neighboring transformations to agree on a common joint location, and (3) $\mathcal{E}_{\text{weight}}(\mathcal{W})$, which constrains the weights of neighboring points to be the same. With coefficients α, β, γ for each term, we write the entire objective as

$$\underset{\mathcal{T},\mathcal{W}}{\text{argmin}} \quad \alpha\,\mathcal{E}_{\text{fit}}(\mathcal{T},\mathcal{W}) \;+\; \beta\,\mathcal{E}_{\text{joint}}(\mathcal{T}) \;+\; \gamma\,\mathcal{E}_{\text{weight}}(\mathcal{W}). \qquad (1)$$

Fitting Objective $\mathcal{E}_{\text{fit}}$. This term measures the alignment distance of the DSG to all frames. For each sample, we measure its alignment distance to other frames by transforming it to the frame and finding the closest corresponding point. Given the closest corresponding point, the total alignment distance is given by the formula

$$\mathcal{E}_{\text{fit}}(\mathcal{T},\mathcal{W}) = \sum_{\mathbf{x}} \sum_{F_g} d\left(T_{j(\mathbf{x})}^{(f(\mathbf{x})\to\text{Ref})}(\mathbf{x}), T_{j(\mathbf{x})}^{(g\to\text{Ref})}\left(\mathbf{y}_{j(\mathbf{x})}^{(g)}\right)\right). \qquad (2)$$

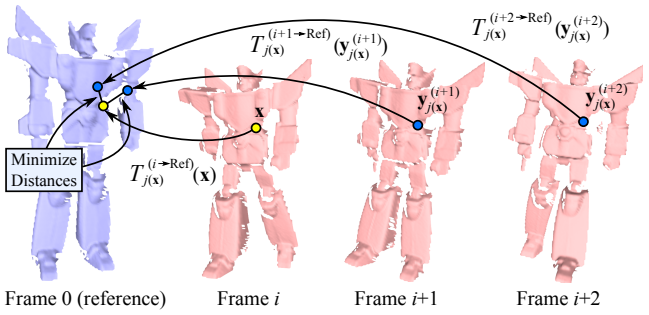


Fig. 5. To measure alignment, we compute distances between samples (yellow) and their closest corresponding points (blue) on the reference frame. Adding up these distances measures the alignment of all frames. We optimize for the transformations and weights that minimize this total distance.

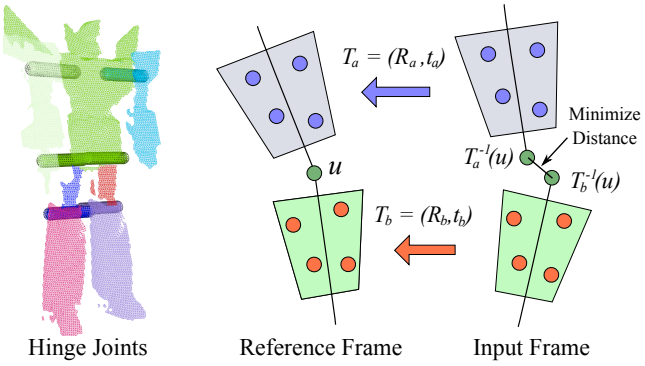


Fig. 6. (Left) Hinge joints (shown as bars) are estimated automatically. (Middle & Right) The joint constraint constrains the distance between transformed joint locations so that the joint stays intact.

Here, F_g denotes the frame g, $d(\cdot,\cdot)$ is a distance measure, $\mathbf{x}$ is the position of each sample, $f(\mathbf{x})$ is the index of the frame from which the sample was selected, $j(\mathbf{x})$ is the index of the transformation assigned (as a weight) to the sample, and $\mathbf{y}^{(g)}_{j(\mathbf{x})}$ is the sample's closest corresponding point in frame g when transformed according to transformation $j(\mathbf{x})$. This formula computes the distance between the sample its corresponding point, but both points have been transformed to the reference frame (illustrated in Figure 5). The resulting values are summed up over all sample positions and all frames g to compute the total alignment distance.

We design the distance $d(\mathbf{x},\mathbf{y})$ to be robust under missing data and outliers. This distance measure first determines whether the correspondence is valid according to three criteria. If it passes, we include their distance in $\mathcal{E}_{\text{fit}}$; otherwise their distance is 0 and not included in the term. We compute d using the formula

$$d(\mathbf{x},\mathbf{y}) = \begin{cases} \eta_{\text{pt}}\,\|\mathbf{x}-\mathbf{y}\|^2 + \eta_{\text{pl}}\,\left((\mathbf{x}-\mathbf{y})\cdot\vec{\mathbf{n}}_{\mathbf{y}}\right)^2 & \text{if } \mathbf{x},\mathbf{y} \text{ is valid} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Here, $\mathbf{x}$ is the sample, $\mathbf{y}$ is its corresponding point, and $\vec{\mathbf{n}}_{\mathbf{y}}$ is the surface normal of $\mathbf{y}$ also transformed to the reference frame. We use the weights $\eta_{\text{pt}} = 0.2$ and $\eta_{\text{pl}} = 0.8$ for our experiments.

The three criteria for a correspondence to be valid is as follows.

(1) Assume that the sample $\mathbf{x}$ is from frame $f(\mathbf{x})$. If $\mathbf{y}$ is from frame g that was added before frame $f(\mathbf{x})$, i.e., when $g < f(\mathbf{x})$, then $\mathbf{y}$ is automatically invalid. To see this, whenever we add a new frame, we add samples to the DSG only from surface parts that are missing in all previous frames (for details see Section 6). Therefore, for $g < f(\mathbf{x})$, no sample added from frame $f(\mathbf{x})$ will have a corresponding point in any frame g.

(2) Because of scanner occlusion, a sample $\mathbf{x}$ from frame f may not have a corresponding point in frame g even when $g > f$. Adapting the strategy from Pekelny and Gotsman [2008], we use thresholding to detect this case. The corresponding point $\mathbf{y}$ is invalid when

—the Euclidean distance $\|\mathbf{x}-\mathbf{y}\|$ exceeds a threshold τ_d,
—the angle between their normals exceeds a threshold τ_n,
—or the distance exceeds a smaller threshold τ_b when $\mathbf{y}$ lies on the boundary of F_g.

(3) When we optimize the weights (see Section 5.3), the weight value $j(\mathbf{x})$ itself becomes a variable, not a fixed constant. Therefore, we compute a separate closest point $\mathbf{y}^{(g)}_a$ for each potential transformation $T^{(f\to g)}_a(\mathbf{x})$. Now, consider the *current* weight $j(\mathbf{x})$ assigned to the sample point. If the closest point for this current weight is invalid, we set all closest points $\mathbf{y}^{(g)}_a$ invalid for all transformations a.

The reason for this strategy is that, if the closest point for the *current* weight is invalid, then the corresponding surface for that sample is most likely missing in frame g. However, by coincidence one of the other transformations may move $\mathbf{x}$ very close to the range scan, but to a completely wrong location. As a result, the weight optimization will prefer to assign this incorrect transformation. Our strategy prevents this problem by conservatively declaring that there is no valid corresponding point for any transformation if the transformation of the current weight does not yield a valid correspondence.

Joint Objective $\mathcal{E}_{\text{joint}}$. The joint term constrains neighboring transformations to agree on a common joint location. It ensures that the parts stay connected to each other and do not drift apart. We support automatically detecting and constraining two types of joints: 3 DOF ball joints and 1 DOF hinge joints. First, we will explain the definition of the joint constraints. Then, we will explain how the joint parameters are computed.

We define the joint locations in the reference frame. A hinge joint specifies that two transformations are connected along a *hinge axis*, which means that both transformations transform the axis to exactly the same location. An example of hinge joints detected for the robot model is illustrated in Figure 6 (left). A ball joint says that the transformations connect on a single point $\mathbf{u} \in \mathbb{R}^3$.

Once we know these joint locations and types, we can constrain the transformations to map the joint locations to the same place (Figure 6, middle & right). We constrain the joints using the formula

$$\mathcal{E}_{\text{joint}}(\mathcal{T}) = \sum_{\text{All } F_i} \sum_{\substack{\text{Valid Joints} \\ (a,b)}} \sum_{t\,\in\,[-10s..10s]} \left\| T_a^{(i\to\text{Ref})^{-1}}(\mathbf{u}_{ab} + t\vec{\mathbf{v}}_{ab}) - T_b^{(i\to\text{Ref})^{-1}}(\mathbf{u}_{ab} + t\vec{\mathbf{v}}_{ab}) \right\|^2. \tag{4}$$

Here, $\mathbf{u}+t\vec{\mathbf{v}}$ is used as the parametric form for the hinge joint, with $\mathbf{u}$ as the location of the joint, $\mathbf{v}$ as the direction of the hinge axis, and a scalar t to generate points along the hinge axis. Since a joint is a pairwise relation, joint parameters are specified for pairs of transformations (a,b) using the notation $\mathbf{u}_{ab}$ and $\vec{\mathbf{v}}_{ab}$). For a hinge

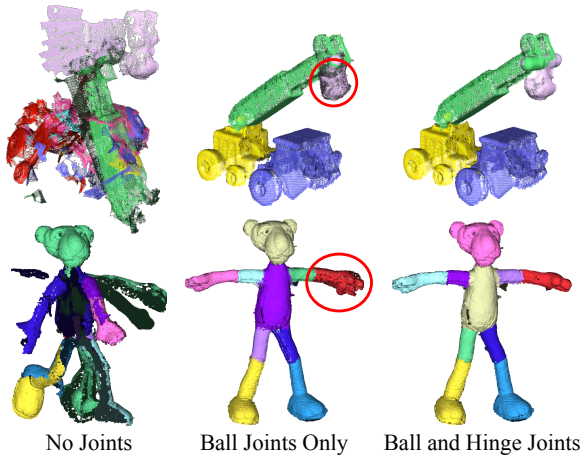


Fig. 7. (Left) Removing the joint constraint causes the registration to fail. (Middle) Using only ball joints, we obtain an acceptable registration. However, there are too many degrees of freedom, causing slight misregistrations (red ovals). (Right) Using both ball and hinge joints produces the most accurate registration.

joint, this formula constrains a set of 20 points along the hinge axis using t in the range of $[-10s..10s]$ and the point spacing s of the range scanning grid [Knoop et al. 2005]. In the case of a ball joint, we set $\vec{\mathbf{v}}_{ab} = 0$, so this term constrains only one point $\mathbf{u}_{ab}$. Inverses of the transformations are used in this term because the joint locations are defined on the reference frame.

Figure 7 illustrates the benefit of using the joint constraint in the optimization. The joints help to preserve the connectivity between each part. In addition, restricting the degrees of freedom appropriately using the hinge joints helps the optimization to converge to the correct local minimum.

Detecting Joint Locations. To detect joints and estimate their locations, we first find which pairs of transformations (a, b) are likely to share a joint in between, and then we determine the location using the transformation values that we have solved for each frame.

We use the edges of the DSG to find pairs of transformations (a, b) likely to have a joint. We restrict our attention to *boundary edges:* edges that have different weights assigned to each end. If there are many boundary edges in the DSG where one end has weight corresponding to transformation a and the other end with weight b, then these transformations neighbor each other and are likely to share a joint in between. On the other hand, if there are no such edges, then most likely there is not a joint between these transformations. To help our discussion, let a boundary edge in the DSG be *incident to transformation* a if one of its end points $\mathbf{x} \in S$ has weight $j(\mathbf{x}) = a$. If either of the following ratios exceeds a threshold (set to 15%):

$$\frac{\#\text{ edges incident to both } a, b}{\#\text{ bdy edges incident to } a}, \quad \frac{\#\text{ edges incident to both } a, b}{\#\text{ bdy edges incident to } b} \tag{5}$$

we take the pair a, b as a candidate for sharing a joint. We then average all endpoints of edges incident to both a, b to obtain an estimate $\mathbf{u}_{\text{est}} \in \mathbb{R}^3$ of the joint location. Since we will define joint locations on the reference frame, we compute $\mathbf{u}_{\text{est}}$ on the reference frame by transforming the DSG samples to the reference. This estimate of the joint location serves to regularize the optimization below and to prune unreasonable estimates of the joint location.

Once we have a set of candidate pairs (a, b) and estimated joint locations $\mathbf{u}_{\text{est}}$, we solve for joint locations $\mathbf{u}$ on the reference frame based on the solved transformations. We perform a least-squares minimization for each pair (a, b) [Cheung et al. 2003]:

$$\underset{\mathbf{u}\in\mathbb{R}^3}{\operatorname{argmin}} \sum_{\text{All } F_i} \left\| T_a^{(i\to\text{Ref})^{-1}}(\mathbf{u}) - T_b^{(i\to\text{Ref})^{-1}}(\mathbf{u}) \right\|^2 + \lambda \left\| \mathbf{u} - \mathbf{u}_{\text{est}} \right\|^2 \tag{6}$$

The first term aims to find the location $\mathbf{u}$ that stays fixed under the transformations, and the second term helps to pull the location closer to $\mathbf{u}_{\text{est}}$ in case the joint is close to being a hinge and admits multiple solutions.

We first try to detect hinge joints using this minimization. We initially set $\lambda = 0$ and solve the least-squares problem using the SVD. The joint is a hinge if the ratio of the smallest singular value to the sum of the singular values is less than a threshold (set to 0.1). If this is the case, we truncate the smallest singular value to zero and solve for the equation of the line $\mathbf{u}' + t\vec{\mathbf{v}}'$ satisfying the system. The final hinge joint parameter $\mathbf{u}$ is the point on this line that is closest to $\mathbf{u}_{\text{est}}$, and $\vec{\mathbf{v}}$ is the normalized line direction $\vec{\mathbf{v}}'/\|\vec{\mathbf{v}}'\|$.

If the joint is not a hinge, it is a ball joint and we determine a single joint location $\mathbf{u}$ with $\vec{\mathbf{v}} = 0$. In this case, we solve the minimization once again, but with $\lambda = 0.1$ to pull the solution nearby the estimated location.

Finally, we perform a sanity check after solving for the joint location. If the distance between the solved and estimated locations exceeds the length of a hinge ($\|\mathbf{u} - \mathbf{u}_{\text{est}}\| > 20s$), we consider it an unreasonable estimate and discard the joint.

Weight Objective $\mathcal{E}_{\text{weight}}$. Constraining the solution to solve for binary weights transforms the problem into a discrete labeling problem, where we try to find an optimal assignment of transformations to the sample points $\mathbf{x} \in S$. The goal of the weight objective is to constrain neighboring samples to have a similar weight. This way, sets of samples with the same weight form well-connected and contiguous regions on the DSG.

We use a simple constant penalty when two neighboring weights are different:

$$\mathcal{E}_{\text{weight}}(\mathcal{W}) = \sum_{(\mathbf{x},\mathbf{y})\in E} I\left(j(\mathbf{x}) \neq j(\mathbf{y})\right), \tag{7}$$

where E is the set of all edges in the DSG, and $I(\cdot)$ is 1 if the argument is true and 0 otherwise. This is known as the Potts model, a discontinuity-preserving interaction term widely used for labeling problems [Boykov et al. 2001].

5.3 Optimization

To perform the optimization, we divide the solver into two phases and alternate between each phase until the solution converges (see Algorithm 2). In the first phase, we keep the weights fixed and solve for the transformations (lines 6-10), and in the second phase, we keep the transformations fixed and solve for the weights (lines 16-22). This strategy works well in practice and produces a good alignment within a few iterations. Also, we try to detect if previously disappeared parts have reappeared in the new frame (line 12).

In our experiments, we observed that the transformations for a frame do not change much after the frame is first introduced and optimized. Therefore, we solve for the transformations only on the newest c frames that have been optimized. We can think of this as a "sliding window" in which to optimize the transformations. Lowering the value of c improves the speed of the registration, while raising this value may produce a more accurate registration at the

Algorithm 2: OPTIMIZE $\mathcal{T}, \mathcal{W}$ $(DSG, \mathcal{T}, \mathcal{W}, F_0, \ldots, F_{i+1})$

Data: Dynamic sample graph (DSG), associated weights $\mathcal{W}$, transformations for all frames $\mathcal{T}$, and all initialized input frames $F_0, \ldots, F_{i+1}$
Result: Optimized transformations and weights $\mathcal{T}, \mathcal{W}$

1 **begin**
2 Select a subset of frames to optimize the transformations (e.g. a sliding window of 1–10 frames);
3 **while** *Not converged* **do**
4 **begin** (**Phase 1:** Solve for the transformations $\mathcal{T}$)
5 Re-estimate joint locations and types;
6 **while** *Not converged* **do**
7 Update the closest points $\mathbf{y}_{j(\mathbf{x})}^{(g)}$ for all $\mathbf{x} \in S$ and all F_g, where $g \in [0\,..\,i{+}1]$;
8 Construct the sparse matrices for $\mathcal{E}_{\text{fit}}$ and $\mathcal{E}_{\text{joint}}$;
9 Solve linear system and update transformations;
10 Check convergence criteria;
11 **end**
12 Detect reappearing transformations in F_{i+1} by aligning occluded parts with unmatched surface points (Section 7);
13 Check convergence criteria;
14 **if** converged **then break**;
15 **begin** (**Phase 2:** Solve for the weights $\mathcal{W}$)
16 Update the closest points $\mathbf{y}_a^{(g)}$ for all samples $\mathbf{x}$, all F_g, and all transformations a, where $g \in [0\,..\,i{+}1]$ and $a \in [0\,..\,B-1]$;
17 Precompute $\mathcal{E}_{\text{fit}}$ for all $\mathbf{x}$ and for all a;
18 Create $\mathcal{E}_{\text{weight}}$ using the edges of the DSG;
19 Solve discrete labeling using α-expansion;
20 Discard parts that are too small;
21 Reuse unassigned weight components by splitting regions with highest $\mathcal{E}_{\text{fit}}$ error;
22 Update the weights for each sample $\mathbf{x}$;
23 **end**
24 **end**

cost of speed. Note that this only affects optimizing the transformations; the weights are always optimized using all frames.

Phase 1: Optimizing the Transformations. For optimizing the first phase, we solve for the transformations minimizing the terms $\alpha\ \mathcal{E}_{\text{fit}} + \beta\ \mathcal{E}_{\text{joint}}$ from Equation 1, while keeping the weights fixed. Since the location of the closest corresponding points depend on the transformations, we use an iterative approach in the spirit of the iterative closest point (ICP) algorithm [Besl and McKay 1992] (line 6–10 in Algorithm 2). We first keep the transformations fixed and compute the closest points, then we keep the corresponding points fixed and optimize the transformations, and we repeat this alternation until convergence.

We perform the optimization using the Gauss-Newton algorithm, linearizing the objective function in each iteration by substituting a linearized form of each rigid transformation. To solve for the transformations on a limited number of frames, we can simply remove the variables/constraints (and also not update closest points) involving transformations from frames outside of the set of interest. This significantly reduces the time to perform this phase.

Phase 2: Optimizing the Weights. For the second phase, we solve for the weights of each sample point $\mathbf{x}$ that minimize the terms $\alpha\ \mathcal{E}_{\text{fit}} + \gamma\ \mathcal{E}_{\text{weight}}$ from Equation 1, while keeping the transformations fixed. Since we constrain the weights to be binary, we are essentially solving for the assignment of transformations for each sample point that minimizes the total error. We solve this discrete optimization problem using the α-expansion algorithm [Boykov et al. 2001; Boykov and Kolmogorov 2004; Kolmogorov and Zabih 2004]. Here, the edges of the DSG directly specify smoothness constraints between sample points. To save computation time during the optimization, we precompute $\mathcal{E}_{\text{fit}}$ in the DSG and store the values in a 2-dimensional hash table for quick access. Specifically, we precompute and store the summand $d(\mathbf{x}, \mathbf{y})$ of $\mathcal{E}_{\text{fit}}$ separately per sample $\mathbf{x}$ and per transformation a, summed over all frames g.

The initial values of the weights are taken from the initial pairwise registration (Section 6). When there is no initial pairwise registration available, the weights are initialized so that the entire scan is a single rigid part. A splitting strategy (described next) automatically divides the model into multiple parts.

After the optimization, it may be the case that some transformations are applied to too few samples of the DSG. If the number of points for a rigid part is less than 5 or 1% of the total number of sample points, then we remove the rigid part from the DSG and replace the weight of its points with the weight of the closest point from a different part. This results in "unused" transformations that are not assigned to any samples.

Instead of just throwing away these unused transformations, we can reintroduce them in a different location to reduce registration error. We split the region with the highest registration error in half and introduce the unused transformation by replacing the weights in one of these halves [Chang and Zwicker 2009]. This adds more degrees of freedom, allowing the optimization to refine the alignment further for the region.

Specifically, we compute an average registration error for each rigid part by averaging the fitting objective (Equation 2) for the points of each part separately. We then split the part with the highest error by randomly selecting two seed points and dividing according to which seed is closer. We leave one of the new parts as is, but we replace the weights of the points in the other with an unused transformation. This splitting process is continued until the highest registration error is below a threshold $0.4s$, or there are no unused transformations left.

Checking for Convergence. We perform a convergence check (Algorithm 2, lines 13-14) after solving for the transformations. To detect if the optimization for the transformations has converged, we monitor the change of the objective function by examining the value of the minimized residual. Denoting the total error residual at iteration ι as E_ι, we apply the criterion $|E_\iota - E_{\iota+1}| < \epsilon(1 + E_\iota)$ (where $\epsilon = 1.0 \times 10^{-6}$) and stop the optimization immediately if this condition is met. In our experiments, we observed that in most cases the optimization converges in about 3–5 iterations. However, the optimization may enter an oscillating mode, where the closest points switch back and forth indefinitely between a few points. Because of this, convergence is not guaranteed; so we limit the maximum number of iterations to 7 in practice. Despite the lack of convergence, we have not encountered any major problems in practice.

6. MAINTAINING THE DYNAMIC SAMPLE GRAPH

The dynamic sample graph (DSG) is an important component that is involved in all stages of our algorithm. In this section, we discuss the details about how we manage the DSG, including how to transfer the initial pairwise registration (Section 4) into a format compatible with the DSG, how to update the DSG with new samples when a frame is added, and how to interpolate the sparse weight function

defined on the vertices of the DSG. Note that precomputing the DSG sample candidates happens only once in the entire algorithm (line 3, Algorithm 1). Applying the initial pairwise registration is a completely separate event that happens before each global registration (line 6, Algorithm 1). Finally, updating the DSG happens after each global registration (line 9, Algorithm 1).

DSG Sample Candidates. The samples of the DSG are taken from points in each frame. To determine the DSG samples efficiently, for each frame f we precompute a random set of candidate points to be included in the DSG. To help our discussion, let us use the symbol U_f to denote the set of candidate points in frame f.

To precompute U_f for each frame f, we find an approximately "maximal" sampling of each frame based on a sample distance τ_s specified by the user. The goal of this maximal sampling is to sample the points of frame f so that there are no more points in the scan farther than τ_s to the sampled subset. Thus, τ_s controls the resolution (or point density) in the DSG.

We use a modified version of the best-candidate sampling algorithm [Mitchell 1991] to generate a maximal sampling according to τ_s. To do this, we keep track of the average distance of the best candidate over the last 100 trials, and we keep adding samples until the averaged distance falls below $0.57\tau_s$ (the constant was determined empirically). Although this is only approximate, it is simple to implement and much faster than checking the "maximal" criteria every time we add a sample.

Applying the Initial Pairwise Registration. Each time we introduce a new frame into the global registration, we need to find an initial value for the transformation of each rigid part of the DSG. We can use the initial pairwise registration between the new frame and the previous frame for this purpose. However, we need to translate the alignment information so that it applies to the rigid parts of the DSG instead of the points of the scan.

For each rigid part of the DSG, we determine its initial transformation by blending the corresponding transformations from the initial pairwise registration. For each sample point in the rigid part, we find the closest point in the previous frame and store the transformation assigned to that point in the initial registration. This results in a list of transformations for the rigid part. Then, we uniformly blend all transformations in the list using Dual Quaternion Linear Blending (DLB) [Kavan et al. 2008] to produce a single initial transformation. Although the blending produces a slightly different transformation than the original values, this strategy worked well in practice.

In the case of the first two frames, the DSG has just been created and the samples do not have any weights. At this point the DSG is exactly a subset of the reference frame, so we directly copy the transformations from the initial registration, while limiting the total number of unique transformations to the maximum B.

Updating the DSG. At the very beginning of the algorithm, the initial set of samples in the DSG is exactly U_{Ref}. However, after we finish optimizing each new frame in the global registration, we update the DSG to incorporate more surface data from the new frame and to reflect changes in the registration. During this process, we select the samples so that they give uniform and adequate coverage of the entire surface seen so far. To do this effectively, we actually resample the DSG from scratch in each update. This is because the global registration changes the alignment of all frames, and certain samples that were not redundant before may become redundant and need to be removed.

To start, we create a new and empty DSG and initialize its samples to U_{Ref}. Then, for each frame f, we add points from the set of sampled points U_f to the DSG that (1) do not overlap with the points added so far and (2) have a valid weight interpolated from the old DSG [Pekelny and Gotsman 2008].

First, to decide overlap, we transform the current points in the (new) DSG to frame f. Then, a point from U_f overlaps with the DSG if the distance to the closest DSG point is less than τ_s. To make this more robust to registration error, if the surface normals of the two points differ by less than 90°, we first project the point (from U_f) onto the plane of the DSG point (using the DSG point's surface normal) and then compute the distance. This is useful to prevent forming multiple layers of the surface, which may occur due to registration error.

Second, to interpolate the weight values from the old DSG, we first transform the old DSG to frame f and divide it into rigid parts. To determine the weight for a point in the new DSG, we compute the closest distance between the point and each rigid part (of the old DSG). We convert these distances to scores by normalizing them to sum to 1. The weight of the point is the transformation of the part with the highest score, but only if the highest score is greater than three times the upper quartile (median of the largest half) of all scores. Otherwise, we consider it an ambiguous case, and a valid weight cannot be determined for the point. Also, the weight is invalid if the highest scoring transformation is marked as occluded for frame f (more details about occlusion in Section 7).

After the resampling is complete, we form edges on the new DSG. We transform it to the reference frame and compute the k-nearest neighbor graph of its samples ($k = 15$). To prevent undesired edges between separate (but spatially near) parts, we discard edges that stretch in length more than twice when transformed to each frame. However, if the edge is between parts that share a joint, we do not discard the edge. This is because discarding these "joint edges" may bias the discrete labeling optimization and cause the boundary between parts to get "stuck" in particular locations. After forming the edges, we finally discard the old DSG and replace it with the new one. This concludes the updating process for the DSG.

7. HANDLING OCCLUSION

In range scan sequences, it is often the case that large parts of the object are occluded in some scans and reappear later. We detect if rigid parts are completely occluded in a frame, adjust the optimization procedure to deal with missing parts, and detect when parts reappear to reintroduce them into the optimization.

Detecting Occluded Parts. When a part of the surface is partially or completely occluded in a frame, the transformation for this part may have few or no valid correspondences constraining it in the optimization. In these cases, it may not be possible to solve for the rigid transformation of that part. In our algorithm, we automatically detect this and exclude these parts from the optimization (line 7 of Algorithm 1).

First, we update the closest corresponding points for each DSG sample and determine their validity according to the three criteria of the robust distance measure (Equation 3). As before, let us partition the DSG into its rigid parts. For each rigid part, if the number of points in the part that have a valid corresponding point falls below a threshold (5, or 10% of the part), then we mark the part's transformation as "occluded" for the new frame. The corresponding rigid part is considered "occluded" as well.

Excluding Occluded Transformations in the Optimization. We need to handle occluded transformations and parts when we solve for the transformations (Phase 1) and weights (Phase 2) in Algorithm 2. In Phase 1, we do not solve for occluded transformations.

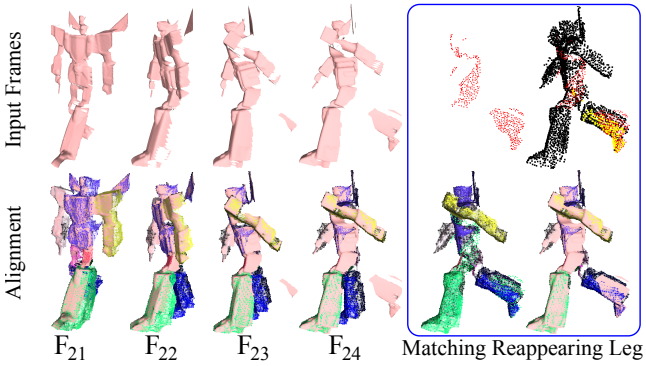


Fig. 8. Matching a reappearing leg. (Left) The right leg disappears in frame 22 and 23, and it reappears in a different location in frame 24. The top row shows the input frames, and the bottom row shows the alignment performed without handling the reappearing leg. (Right Square) The top left shows the points in frame 24 that are unmatched by the DSG. We align the occluded parts to these unmatched points. The top right highlights the correspondences found by matching the DSG with the unmatched points. Finally, the bottom row shows the alignment result, shown with unmatched points (left) and input frame (right).

Instead, we substitute a value computed based on the joint constraints with neighboring parts. If there are no neighbors, we use the value from the last frame; if there is exactly one, we apply the relative transformation to the neighbor derived from the last unoccluded frame; and if there are two or more, we solve for the transformation that best fits all joint constraints [Pekelny and Gotsman 2008].

While we can simply exclude transformations in Phase 1, we cannot do the same for optimizing the weights in Phase 2 because we are always optimizing the weights on all frames simultaneously. The problem is that every DSG sample $\mathbf{x}$ needs a distance error value $d(\mathbf{x}, \mathbf{y})$ for all transformations and all frames. A reasonable error value is needed for all samples when a transformation is occluded in a certain frame.

For the occluded transformation's error, we use the error value of the current transformation $j(\mathbf{x})$ for each sample $\mathbf{x}$. If transformation $j(\mathbf{x})$ is also occluded, we use the minimum error value among all non-occluded transformations. If we had used a zero error, the optimization would tend to assign the occluded transformation to the samples. Also, too high a value would cause the optimization to not assign the occluded transformation to any samples at all. Our strategy takes the middle ground and works well in practice.

Detecting Reappearing Parts. When an occluded part suddenly reappears in a new frame, we need to start tracking it again. Otherwise, the algorithm could mistakenly treat it as new surface geometry, thus duplicating the part multiple times in the reconstruction. We detect and handle this case during the global registration (line 12 of Algorithm 2). Note that detecting reappearing parts cannot be handled by our initial pairwise registration, because it can only align parts not occluded in *both* the source and target.

To detect if an occluded part is reappearing in the new frame $i+1$, we first transform the DSG to frame $i+1$ and determine the weights of the frame's sample points U_{i+1} using the weight interpolation in a separate step (discussed in Section 6). If a valid weight cannot be determined (i.e. it is an ambiguous case where the highest score is less than three times the upper quartile of all scores, or the highest scoring transformation is occluded), then we consider the point to be "unmatched" and add it to a set M of unmatched points. If there is a sufficient number of unmatched points ($|M| > 0.05|U_{i+1}|$), we interpret this as an indication that a previously occluded part is reappearing. Therefore, we attempt to match these points by aligning them with transformations that were previously marked as occluded for frame $i+1$.

Fig. 9. Reconstruction results for the Robot dataset.

Here, we use the same procedure as Phase 1 of Algorithm 2 to solve for the values of the occluded transformations (Section 5.3). However, we expect unmatched points to be far away from the current position of the DSG. Thus, we adjust the optimization where

—we only solve for the values of the occluded transformations by aligning occluded parts of frame $i+1$ to M,
—we set thresholds to higher values: $\tau_d = 50s$ and $\tau_n = 45°$,
—and we increase the weight of the joint constraint to $\beta = 1000$.

These modifications ensure that we obtain an alignment even when the points in M are far away from the DSG, while the higher value of β prevents the optimization from falling into local minima. Figure 8 shows an example where a reappearing part in the robot sequence is detected and registered. After solving for the occluded transformations, we run the occlusion detection routine once more to update each transformation's occlusion status, and then we continue on with the optimization.

8. EXPERIMENTAL RESULTS

8.1 Reconstruction

We implemented our algorithm in C++ and tested it with several real-world and synthetic datasets exhibiting articulated motion. After we have aligned all frames, we reconstruct a triangle mesh from a dense sampling of the DSG. We use the streaming wavelet surface reconstruction algorithm by Manson et al. [2008].

The car and robot datasets were acquired by Pekelny and Gotsman [2008] using a Vialux Z-Snapper depth camera. These sequences were created by animating the physical model while capturing each frame from a different viewpoint. Each sequence has 90 frames, and consists of 4 and 7 parts, respectively. The results are shown in Figures 9 and 10. The top row shows some of the input frames in the sequence. Notice that there is a significant amount of occlusion in some of the frames. The middle row shows the recon-

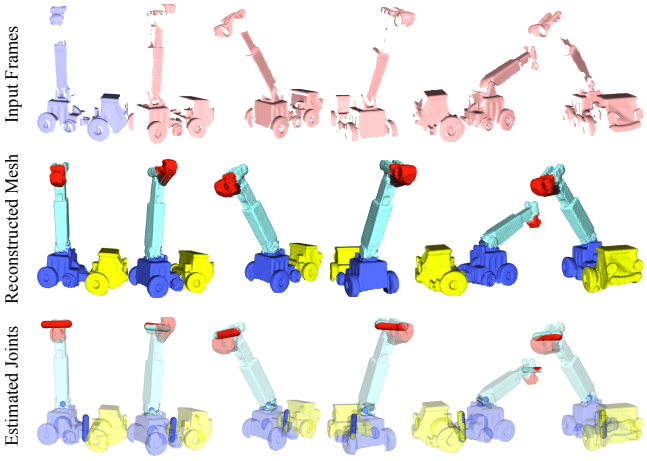


Fig. 10. Reconstruction results for the Car dataset.

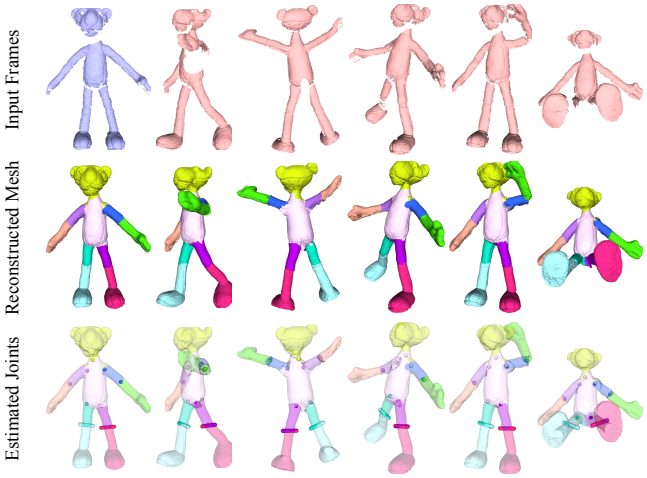


Fig. 11. Reconstruction results for the Pink Panther dataset with faster input motion.

structed mesh using the algorithm, with weights obtained by interpolating the weights on the sample set. The bottom row shows the estimated joint locations, where hinge joints are represented by a short bar and ball joints by a sphere. Both the reconstruction results and the weight estimation are faithful to the input data.

To test our algorithm on a more deformable subject, we acquired two range scan sequences of a furry pink panther toy using a Konica Minolta VI-910 laser scanner. We animated one sequence with a slower motion, and another sequence with a faster motion. Reconstruction results are shown in Figure 11. Although the furry texture caused noise on the scanned surface, we obtained a reasonable reconstruction of both the surface geometry and the weights.

Finally, we tested our algorithm on synthetic depth scans of a walking man, where the camera is rotating around the subject. To test the effect of occlusion in our algorithm, we captured two sequences, one using a single virtual camera, and the other using two virtual cameras 90° apart. The reconstruction results are shown in Figure 12. The results from both datasets are reasonable, but the first sequence was less successful due to the large amount of occlusion of the arms. With two virtual cameras, we obtained a better reconstruction that was able to reproduce the fine detail of the hands.

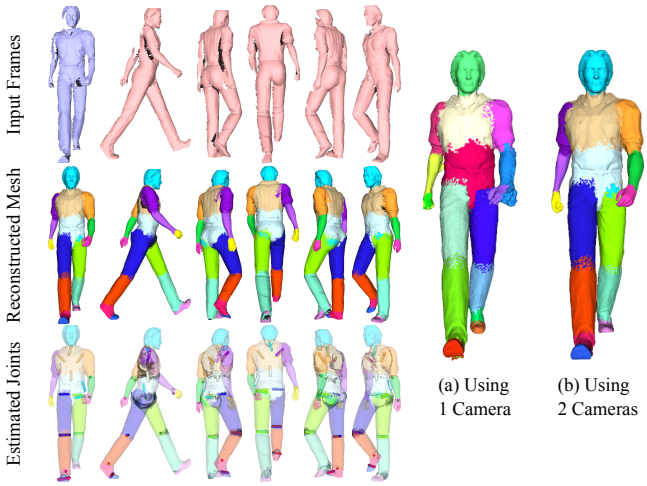


Fig. 12. Reconstruction results for the synthetic Walking Man dataset. On the right, (a) and (b) show a comparison of the reconstruction using scans from one and two virtual cameras.

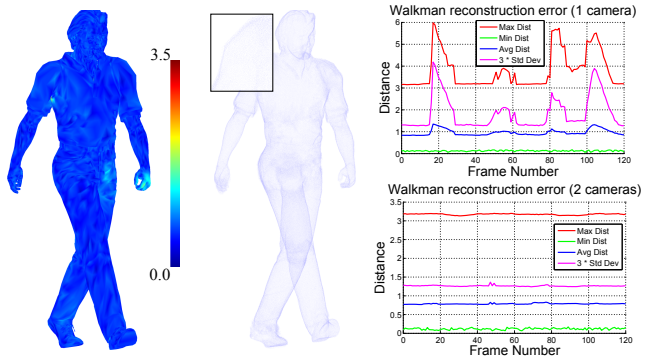


Fig. 13. Reconstructed point cloud and error plot for the walkman dataset. (Left) Visualization of the distance between the ground truth mesh and reconstructed point cloud. The distance is in units of grid sample spacing. (Middle) Noise on the reconstructed surface is caused by registration error. The zoomed-in shoulder region shows points from different surface layers. (Right) Plot of the registration error per frame. The peaks in the top graph correspond to times where the algorithm loses track of the surface (the arms) due to occlusion in the data.

8.2 Validation with Ground Truth Data

To validate our method, we compared our reconstruction results with the original (i.e. ground truth) mesh and animation data that was used to generate the synthetic walkman dataset. Figure 13 (left) visualizes the registration error between the ground truth mesh and the reconstructed point cloud. For each vertex in the ground truth mesh, we computed the distance to the closest point in the point cloud. On the right, the graphs show the min, max, average, and standard deviation of these distances for each frame.

The peaks of the graph in Figure 13 (top right) correspond to frames where the surface is not visible and the algorithm loses track. When the surface is visible (as with the case of 2 cameras), the algorithm is able to track the surface well. The reconstructed

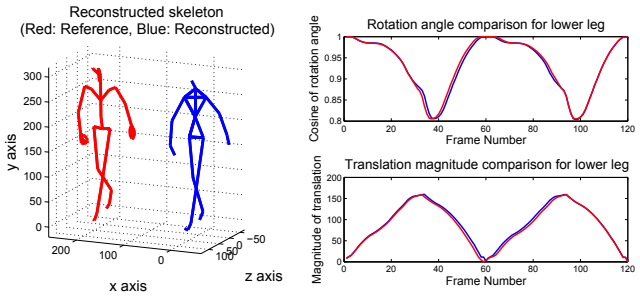


Fig. 14. Comparison between a reconstructed skeleton and the reference skeleton used to animate the synthetic walkman dataset. Both the reconstructed skeleton and transformations closely resemble the reference data.

surface is often noisier than the scan data. This is because the surfaces are not registered perfectly, causing the points to be sampled from multiple surface layers (Figure 13 middle). This causes noise in both the point cloud and the reconstructed triangle mesh.

We also compared the quality of the reconstructed animation with the ground truth data. Figure 14 shows a comparison between the ground truth skeleton and the reconstructed skeleton (made using the solved joint locations and relations, discussed in Section 5.2). The plot on the right compares the numerical values of the transformations for the lower right leg. Here, red indicates the ground truth data, and blue indicates the reconstructed data. The results for other parts were similar. The transformations for the lower torso and the head deviated the most because they were segmented differently compared to the ground truth.

8.3 Parameters

The parameters of our algorithm are summarized in Tables I and II, where Table 1 shows the data dependent parameters and Table 2 lists the parameters that have the same value for all datasets. We expressed many parameters relative to the sample spacing s, which is the average distance between neighboring points on the range scanning grid. This parameter can be computed automatically for each dataset. There are many parameters overall, but the user can keep most of them fixed and change only a few parameters to process a new dataset. We summarize all of them in the tables for reference.

For the variable-valued parameters in Table I, we found that the algorithm was insensitive to the value of B, given that an adequate number is given. However, the algorithm was sensitive to the value of the weight of the $\mathcal{E}_{\text{joint}}$ term β, the weight of the $\mathcal{E}_{\text{weight}}$ term γ, and the minimum distance τ_s between samples in the DSG. For these parameters, appropriate values were found through experimentation. We set β based on how the joint constraint was affecting the registration: we relaxed β if neighboring parts seemed too stiff, or strengthened β if parts were drifting apart. For γ, we lowered its value if the weight optimization was consolidating the parts too much, or we raised the value if the weight optimization divided in the DSG into too many small parts. Finally, the value of τ_s controls the resolution of the DSG; this parameter trades slower performance for a small increase in reconstruction accuracy. However, if τ_s is too large, then the reconstruction may fail entirely. In our experiments, we experimented with different parameter settings but did not exhaustively optimize the parameters to give a better result.

For the fixed value parameters in Table II, there were two exceptions where changing the value produced a better result. For the pink panther dataset with faster input motion, we adjusted the weight value β for the term $\mathcal{E}_{\text{joint}}$ to 10 when matching reappearing parts. This was because the value of 1000 was too stiff to correctly align a reappearing part in one frame of the sequence. Also, for the robot dataset, we lowered the minimum average error for splitting a region to $0.2s$. This caused the implementation to split regions more easily, resulting in better convergence of the weight optimization.

Table I. Data dependent parameters. Distances are given in units of world space coordinates.

Description	Car	PP	Popcorn	Robot	Spock	Walking
Sample spacing s	0.0125	1	2	1	0.4	1
Min dist. τ_s in DSG	$2s$	$5s$	$10s$	$2.5s$	$10s$	$2.5s$
Max num. of parts B	7	10	5	7	7	16
$\mathcal{E}_{\text{joint}}$ weight β	1.0	1.0	3.0	1.0	2.0	1.0
$\mathcal{E}_{\text{weight}}$ weight γ	$0.1s$	$0.5s$	$5.0s$	$0.1s$	$1.0s$	$0.1s$

Table II. Fixed value parameters. Distances are given in units of world space coordinates. A number of parameters are specified relative to the sample spacing s.

Description	Value
Matching closest points (Section 5.2)	
Max correspondence distance τ_d	$10s$
Max corresponding normal angle τ_n	45°
Max boundary correspondence distance τ_b	$1s$
Weight for normals when matching closest points	$2s$
Detecting and constraining joints (Section 5.2)	
Min incident edge ratio for detecting joints	0.15
Max distance between estimated and solved joint locations	$20s$
Number of frames required for estimating hinge joints	3
Hinge joint threshold on the singular values	0.1
Number of samples used to constrain hinges	20
Length of hinge joint constraint	$10s$
Optimization (Section 5.3)	
$\mathcal{E}_{\text{fit}}$ weight α	1.0
Sliding window size c for transformation optimization	5
Optimizing weights (Section 5.3)	
Max number of samples for a rigid part to be removed	5
Max fraction of samples for a rigid part to be removed	0.01
Min average error for splitting a region	$0.4s$
Solver control (Section 5.3)	
Max iterations for the optimization	7
Max iterations for solving the transformations	30
DSG parameters (Section 6)	
Reduced τ_s for generating the final shape	$0.5\ \tau_s$
Nearest neighbors k for DSG	15
Maximum stretch factor for pruning an edge	2.0
Maximum length of an edge	$4\ \tau_s$
Occlusion handling parameters (Section 7)	
Max number of correspondences for a rigid part to be occluded	5
Max fraction of correspondences for a rigid part to be occluded	0.1
Min ratio of remaining geometry to trigger reappearing part match	0.05
Weight for $\mathcal{E}_{\text{joint}}$ for matching reappearing parts	1000
Max correspondence distance for matching reappearing parts	$50s$
Max corresponding normal angle for matching reappearing parts	45°

8.4 Performance

We performed our experiments using a single core of an Intel Xeon 2.5 GHz processor. The timing results are reported in Table III. In the robot and car datasets, the most time-consuming part was the initialization, but in the other cases it was the global registration. The global registration step can execute faster if a smaller sliding window is used, with the trade-off of having a less accurate registration. To improve performance when processing the walkman dataset, we started the registration with a 5-frame sliding window, and after 30 frames we changed to a 1-frame sliding window (denoted as "$5 \rightarrow 1$"). Like other closest point matching algorithms, the most time-consuming part is the closest point computation, which can typically take 30% of the total time. Note that the times in the initialization step reported in Table III do not include pre-

Table III. Performance statistics for our experiments. All the timings are in units of seconds. The bottom row reports the average execution time per frame in each sequence.

Statistic	Robot	Car	PP1	PP2	Walking1	Walking2
Max Bones	7	7	10	10	16	16
Used Bones	7	4	10	10	14	16
Frames	90	90	40	40	121	121
Sliding Window	5	5	5	5	$5 \rightarrow 1$	$5 \rightarrow 1$
Points/Frame	9,391.2	5,387.86	36,683.9	30,003.1	19,843.7	39,699.7
Total Points	845,208	484,907	1,227,356	1,200,125	2,401,082	4,803,662
Samples in DSG	4,970	2,672	4,077	4,203	8,305	8,539
Edges in DSG	37,678	20,707	30,758	31,841	61,711	63,043
Initialization	7,357.68	2,652.57	1,826.27	1,828.98	69.38	134.74
Global Reg	2,287.61	1,200.04	2,184.68	2,624.4	5,574.86	19,789.0
Updating DSG	264.44	117.93	67.90	68.06	876.32	1,617.07
Total Time	9,909.73	3,970.54	4,079.85	4,521.44	6,520.56	21,540.81
Average Time	110.11	44.12	102.00	113.04	53.89	178.02

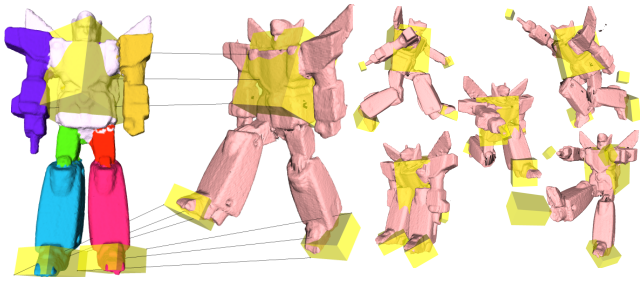


Fig. 15. Reposing the reconstructed robot. Using the solved weights and the hinge joints, we can perform interactive IK on the reconstructed model.

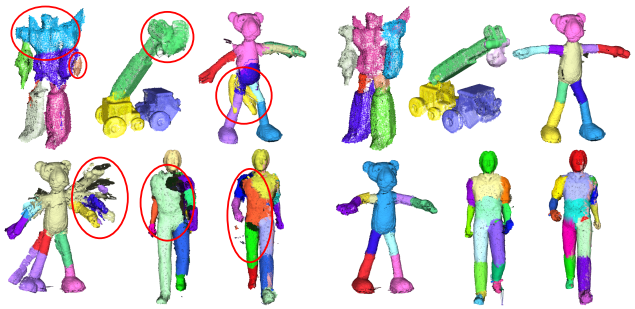
processing time to compute spin images and estimate the principal curvature frame at each vertex.

8.5 Inverse-Kinematics Application

Solving for the weights and joints in the model is useful for reposing and animating the reconstructed model. To demonstrate this, we implemented a tool to perform inverse kinematics on the model. The user can specify point constraints interactively by drawing boxes around a region of interest. Then, the user is able to select one of the constraint boxes and drag it around on the screen to manipulate the model. To perform IK, we use the transformation optimization (Section 5.3) to solve for the rigid transformations of each part that best satisfy the constraints. The details of the optimization are exactly the same as before, except that the joint locations are fixed and the correspondences are given by the user. By running the optimization in a separate background thread, we were able to interactively manipulate the reconstructed model in real-time. Figure 15 shows examples of different poses of the robot created by our system.

8.6 Sequential Registration vs. Simultaneous Registration

To illustrate the benefit of performing simultaneous registration, we compare our algorithm with a sequential registration pipeline. In a sequential registration method, we optimize each frame of the sequence one-by-one, accumulate new samples directly on the reference frame, and discard the frame before moving on to the next. This strategy is essentially a pairwise registration that is applied repeatedly for each frame, because it only performs the registration between the accumulated samples and the current frame.



(a) Results Using Sequential Registration (b) Results Using Simultaneous Registration

Fig. 16. Comparing sequential and simultaneous registration. (a) The sequential strategy gives an unreliable estimate of the articulated structure, because it only uses the movement observed in two frames at a time. This leads to an imprecise registration (marked as red ovals). (b) The simultaneous strategy can correctly estimate the weights that reflect the movement observed in all frames. The registration is more precise, as well as the estimated surface geometry.

The main problem with the sequential registration approach is that it cannot reliably estimate the articulated structure (i.e. weights) based on the movement observed in just two frames at a time. This complicates the situation further for occlusion detection and recovery, which rely on a reliable estimate of the articulated structure.

A comparison between the sequential and simultaneous strategies is shown in Figure 16. Here, we used the two strategies to align each dataset and display the DSG that roughly shows the reconstructed geometry. On the left, we can see that the sequential strategy did not produce the correct weights. As a result, the registration was imprecise, and "extra" surfaces appear where the parts were not aligned properly (for example, on the left arm of the robot). On the right, we show the result using simultaneous registration using the same parameters. The registration is more accurate, and the algorithm produced correct weights that reflect the movement of all frames.

We compared the sliding window strategy with a full global registration strategy on the car and pink panther datasets. Note that the sliding window only applies when solving for the transformations (Phase 1 of Algorithm 2). We always solve simultaneously on all frames when solving for the weights (Phase 2 of Algorithm 2); in this sense our algorithm always performs a "true" global registration regardless of the sliding window size.

Figure 17 shows a comparison of the results using the two strategies. There was not much difference between the results, but the running times suffered significantly with the full global strategy. The running time for the car dataset increases more than it does for the pink panther dataset because it has a greater number of frames.

8.7 Comparison with Wand et al. [2009]

We compare our articulated reconstruction with the non-rigid reconstruction method by Wand et al. [2009]. For the car, robot, and pink panther datasets, their method was not able to reconstruct the entire sequence because there was too much motion between the frames. This is because they rely only on a local optimization using closest points, whereas our method uses a robust initial pairwise initialization that is able to automatically handle frames with large motion. Also, having rigid parts in the optimization helps the reg-

(a) Frame 1 (b) Frame 9 (c) Frame 15 (d) Frame 21

Fig. 19. Registration for a grasping hand sequence [Weise et al. 2007], where the hand starts from an open pose and gradually closes to a grasping pose. Shown are the input data (displayed as a red color mesh) and the sparse DSG. Our algorithm tracks the hand well in the first part of the animation, where most of the surface is visible. In (c), the surface of the fingers start to gradually disappear, and the middle segment of the index finger starts to lose track and rotate backwards. In (d), the algorithm loses track of the middle and ring fingers, because most of these fingers are occluded (except for the fingertips).

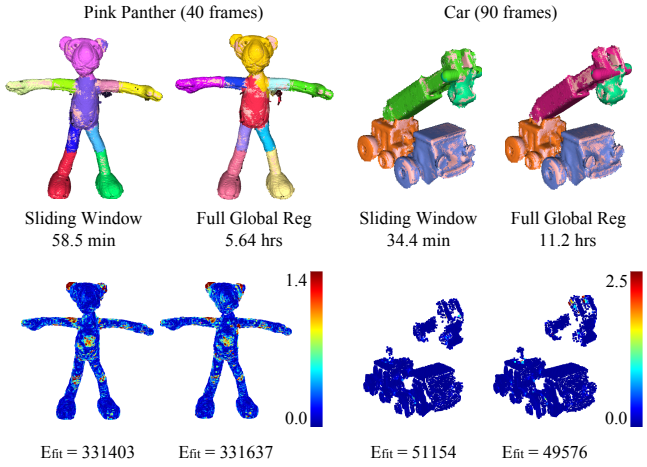


Fig. 17. Comparing transformation optimization with a 5-frame sliding window vs. a full global registration. The bottom row reports the final total value of $\mathcal{E}_{\text{fit}}$ and visualizes the distance from each DSG sample point to an input scan (in units of s^2, squared sample distance).

istration to converge to the correct solution. An example of this is shown in the top row of Figure 18. In this example, an initial pairwise registration between the frames was not used, and the registration was performed as-is. Our method (right) produces a correct registration, while their method (middle) fails for this pair.

We also tested our algorithm on several examples from Wand et al. [2009]. Figure 18 (middle and bottom) shows reconstructions of the hand-2 and popcorn tin datasets, and Figure 19 shows a result for the grasping hand (hand-1) dataset. These sequences exhibit non-rigid motion, especially the popcorn tin dataset. Our algorithm can capture the overall shape and produce a coarse articulated motion of the subject. However, it does not reproduce the fine non-rigid details of the surface deformation.

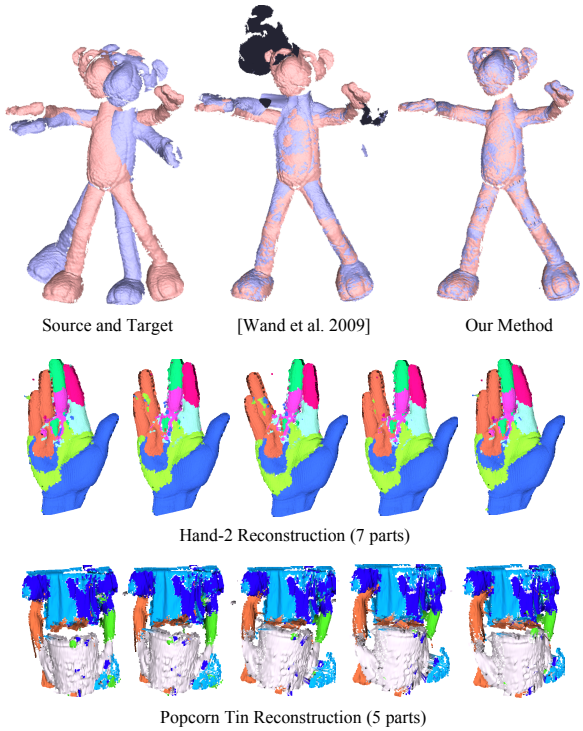


Fig. 18. (Top) Comparison between the non-rigid registration of Wand et al. [2009] and articulated registration on two pink panther frames. While the non-rigid registration fails, the articulated registration succeeds even without an initial pairwise registration between these frames. (Middle and Bottom) Articulated registration on the hand-2 and popcorn tin datasets used by Wand et al. [2009]. Our algorithm is able to produce coarse approximations of the non-rigid motion exhibited in these datasets.

9. SUMMARY AND CONCLUSION

We have presented the articulated global registration algorithm, which reconstructs an articulated 3D model from a set of range scans. We solve for the division of the surface into parts (weights) and the motion for each part (transformations) to align all input scans. For this purpose, we used an improved robust registration to solve for an initial pairwise registration between adjacent frames in the sequence. Then, we formulated a simultaneous registration of all input frames to minimize registration error. This optimization included joint constraints that preserves the connectivity between parts and automatically handled cases when parts are disappear or reappear. We demonstrated that we can reconstruct a full 3D articulated model without relying on markers, a segmentation, or a template. Finally, we demonstrated that the reconstructed model is deformable and can be interactively manipulated into new poses using a simple inverse-kinematics extension of our optimization algorithm.

A limitation of our method is that there needs to be enough overlap between adjacent frames in the range scan sequence to obtain a good alignment. For example, if one frame captures the surface from the front, and the next frame captures the surface from the back, there will be not enough overlap to match these frames together in the registration. This means that the order of the range

scans in the sequence should maintain a reasonable amount of overlap between adjacent pairs of frames. A temporal ordering of the scans, for example, would produce a sequence with a reasonable amount of overlap. However, even this is not enough when there is severe occlusion. For example, our algorithm loses track of the fingers in the hand sequence because of too much missing data, as shown in Figure 19.

Another shortcoming of our ICP-based registration is the handling of "slippable" parts such as cylinders. For example, the fingers of the hand example in Figure 19 have cylindrical symmetry, so the ICP registration can converge into a state where the segments of the fingers are "twisted" or rotated about the axis of symmetry (Figure 19c). Although hinge joints could disambiguate cylindrical symmetries, we found that it was difficult to estimate accurate hinge joints in this case.

Currently our method is applicable for reconstructing articulated subjects and coarsely capturing non-rigid subjects. However, it would be interesting to adapt our algorithm for high-quality non-rigid reconstruction. For this case, estimating "flexible" transformations would be appropriate, for example, estimating affine transformations with additional surface displacements. Also, it would be useful to find a way to optimize for smooth weights without causing overfitting. We believe that there should be a middle ground between solving for a separate transformation for every sample point [Li et al. 2008] and our method of solving for the weight at each sample point.

We would also like to reduce the parameters in our algorithm. An alternative to specifying various thresholds is to use a robust error metric similar to the work of Nishino and Ikeuchi [2002]. In this case, the outliers would automatically be identified during the optimization, without a need to specify hard thresholds. Also, the user currently needs to specify the maximum number of transformations B to approximate the motion. An alternative strategy could have the user specify a maximum alignment error ϵ and change the algorithm to add additional transformations until the alignment error is within ϵ.

Finally, we would like to investigate ways of improving the performance of the algorithm. In particular, since our method estimates the weights and transformations for all frames simultaneously, we need to keep all of the input scans in memory. We would like to develop a streaming version of our algorithm that reduces the memory requirements and allows us to process longer sequences. In addition, if we can detect when reasonable weights have been obtained, we can skip the weight optimization step to save time in the algorithm. We believe that an improved version of our algorithm along these lines can be implemented for real-time markerless motion capture applications.

ACKNOWLEDGMENTS

We thank M. Wand for providing comparisons and useful feedback, S. Buss for helpful discussions, and the members of the UCSD graphics lab for their feedback. We also wish to thank Y. Pekelny and C. Gotsman for sharing the car and robot datasets, T. Weise for the grasping hand dataset, O. Schall for the hand-2 dataset, and P. Fong for the popcorn tin dataset. Additional thanks to G. Debunne for providing the libQGLViewer library, D. M. Mount and S. Arya for the ANN library, Y. Boykov, O. Veksler, R. Zabih for an implementation of their graph cuts algorithm, M. Matsumoto for SFMT, S. Toledo for TAUCS, and J. Manson for surface reconstruction software.

REFERENCES

AHMED, N., THEOBALT, C., DOBREV, P., SEIDEL, H.-P., AND THRUN, S. 2008. Robust fusion of dynamic shape and normal capture for high-quality reconstruction of time-varying geometry. In *CVPR*.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2002. Articulated body deformation from range scan data. *ACM SIGGRAPH 21,* 3, 612–619.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM SIGGRAPH*. 587–594.

ANGUELOV, D., KOLLER, D., PANG, H., SRINIVASAN, P., AND THRUN, S. 2004. Recovering articulated object models from 3d range data. In *Uncertainty in Artificial Intelligence Conference (UAI)*.

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. Scape: shape completion and animation of people. In *ACM SIGGRAPH*. 408–416.

ANGUELOV, D., SRINIVASAN, P., PANG, H.-C., KOLLER, D., THRUN, S., AND DAVIS, J. 2004. The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In *NIPS*.

BESL, P. J. AND MCKAY, H. 1992. A method for registration of 3-d shapes. *IEEE TPAMI 14,* 2, 239–256.

BOYKOV, Y. AND KOLMOGOROV, V. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE TPAMI 26,* 9 (September), 1124–1137.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE TPAMI 23,* 11, 1222–1239.

BRADLEY, D., POPA, T., SHEFFER, A., HEIDRICH, W., AND BOUBEKEUR, T. 2008. Markerless garment capture. *ACM SIGGRAPH*.

CHANG, W. AND ZWICKER, M. 2008. Automatic registration for articulated shapes. *Comput. Graph. Forum (Proceedings of SGP) 27,* 5, 1459–1468.

CHANG, W. AND ZWICKER, M. 2009. Range scan registration using reduced deformable models. *Comput. Graph. Forum (Proceedings of Eurographics) 28,* 2, 447–456.

CHEUNG, G. K. M., BAKER, S., AND KANADE, T. 2003. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *CVPR*. 77–84.

DE AGUIAR, E., STOLL, C., THEOBALT, C., AHMED, N., SEIDEL, H.-P., AND THRUN, S. 2008. Performance capture from sparse multi-view video. *ACM SIGGRAPH*.

DE AGUIAR, E., THEOBALT, C., THRUN, S., AND SEIDEL, H.-P. 2008. Automatic conversion of mesh animations into skeleton-based animations. *Comput. Graph. Forum (Proceedings of Eurographics) 27,* 2, 389–397.

GALL, J., STOLL, C., DE AGUIAR, E., THEOBALT, C., ROSENHAHN, B., AND SEIDEL, H.-P. 2009. Motion capture using joint skeleton tracking and surface estimation. In *CVPR*.

HUANG, Q.-X., ADAMS, B., WICKE, M., AND GUIBAS, L. J. 2008. Non-rigid registration under isometric deformations. *Comput. Graph. Forum (Proceedings of SGP) 27,* 5, 1449–1457.

JAMES, D. L. AND TWIGG, C. D. 2005. Skinning mesh animations. In *ACM SIGGRAPH*.

KAVAN, L., COLLINS, S., ZÁRA, J., AND O'SULLIVAN, C. 2008. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics 27,* 4.

KNOOP, S., VACEK, S., AND DILLMANN, R. 2005. Modeling joint constraints for an articulated 3d human body model with artificial correspondences in icp. In *IEEE-RAS Conference on Humanoid Robots*.

KOLMOGOROV, V. AND ZABIH, R. 2004. What energy functions can be minimized via graph cuts? *IEEE TPAMI 26,* 2, 147–159.

LI, H., ADAMS, B., GUIBAS, L. J., AND PAULY, M. 2009. Robust single view geometry and motion reconstruction. In *SIGGRAPH ASIA*.

LI, H., SUMNER, R. W., AND PAULY, M. 2008. Global correspondence optimization for non-rigid registration of depth scans. *Comput. Graph. Forum (Proceedings of SGP) 27,* 5, 1421–1430.

MANSON, J., PETROVA, G., AND SCHAEFER, S. 2008. Streaming surface reconstruction using wavelets. *Comput. Graph. Forum (Proceedings of SGP) 27,* 5, 1411–1420.

MITCHELL, D. P. 1991. Spectrally optimal sampling for distribution ray tracing. *ACM SIGGRAPH*.

MITRA, N. J., FLÖRY, S., OVSJANIKOV, M., GELFAND, N., GUIBAS, L. J., AND POTTMANN, H. 2007. Dynamic geometry registration. In *Symposium on Geometry Processing*. 173–182.

NEUGEBAUER, P. J. 1997. Reconstruction of real-world objects via simultaneous registration and robust combination of multiple range images. *International Journal of Shape Modeling 3,* 1/2, 71–90.

NISHINO, K. AND IKEUCHI, K. 2002. Robust simultaneous registration of multiple range images. In *ACCV*.

PARK, S. I. AND HODGINS, J. K. 2006. Capturing and animating skin deformation in human motion. *ACM SIGGRAPH*.

PARK, S. I. AND HODGINS, J. K. 2008. Data-driven modeling of skin and muscle deformation. *ACM SIGGRAPH*.

PAULY, M., MITRA, N. J., GIESEN, J., GROSS, M. H., AND GUIBAS, L. J. 2005. Example-based 3d scan completion. In *Symposium on Geometry Processing*. 23–32.

PEKELNY, Y. AND GOTSMAN, C. 2008. Articulated object reconstruction and markerless motion capture from depth video. *Comput. Graph. Forum (Proceedings of Eurographics) 27,* 2, 399–408.

POPA, T., SOUTH-DICKINSON, I., BRADLEY, D., SHEFFER, A., AND HEIDRICH, W. 2010. Globally consistent space-time reconstruction. *Comput. Graph. Forum (Proceedings of SGP) 29,* 5, 1633–1642.

SCHAEFER, S. AND YUKSEL, C. 2007. Example-based skeleton extraction. In *Symposium on Geometry Processing*. 153–162.

SHARF, A., ALCANTARA, D. A., LEWINER, T., GREIF, C., SHEFFER, A., AMENTA, N., AND COHEN-OR, D. 2008. Space-time surface reconstruction using incompressible flow. *ACM SIGGRAPH ASIA*.

SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. In *ACM SIGGRAPH*.

SUMNER, R. W., ZWICKER, M., GOTSMAN, C., AND POPOVIC, J. 2005. Mesh-based inverse kinematics. In *ACM SIGGRAPH*.

SÜSSMUTH, J., WINTER, M., AND GREINER, G. 2008. Reconstructing animated meshes from time-varying point clouds. *Comput. Graph. Forum (Proceedings of SGP) 27,* 5, 1469–1476.

VLASIC, D., BARAN, I., MATUSIK, W., AND POPOVIĆ, J. 2008. Articulated mesh animation from multi-view silhouettes. *ACM SIGGRAPH*.

WAND, M., ADAMS, B., OVSJANIKOV, M., BERNER, A., BOKELOH, M., JENKE, P., GUIBAS, L., SEIDEL, H.-P., AND SCHILLING, A. 2009. Efficient reconstruction of non-rigid shape and motion from real-time 3d scanner data. *ACM Transactions on Graphics 28*.

WEISE, T., LEIBE, B., AND GOOL, L. J. V. 2007. Fast 3d scanning with automatic motion compensation. In *CVPR*.

WEISE, T., LI, H., GOOL, L. J. V., AND PAULY, M. 2009. Face/off: live facial puppetry. In *Symposium on Computer Animation*. 7–16.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. M. 2004. Space-time faces: high resolution capture for modeling and animation. In *ACM SIGGRAPH*.